# PART V

## Functions and Pointers

**Gopher Level**
**Intermediate**

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# PART V

## Functions and Pointers

**Functions**

Reusable

code

blocks

**Pointers**

Stores

Memory

Addresses

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# PART V

## Functions and Pointers

### 🧙‍♂️ Design a Program Using Functions

Rewrite the Log Parser

### ⚙️ Pass By Value Semantics

Passing Structs, Maps and Slices to Funcs

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

Every function starts with a func keyword

```
func {
  // you write your code here...
}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

Every *package-level* function has a name

```
func name {
  // you write your code here...
}
```

*init and main are special names: do not name your funcs using them*

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

This function doesn't declare any input parameters

```
func name() {
  // you write your code here...
}
```

```
func printDefaults() {}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

A function *may* declare *zero or more* input parameters

*(name type, name type)*

```
func name(inputParams) {
  // you write your code here...
}
```

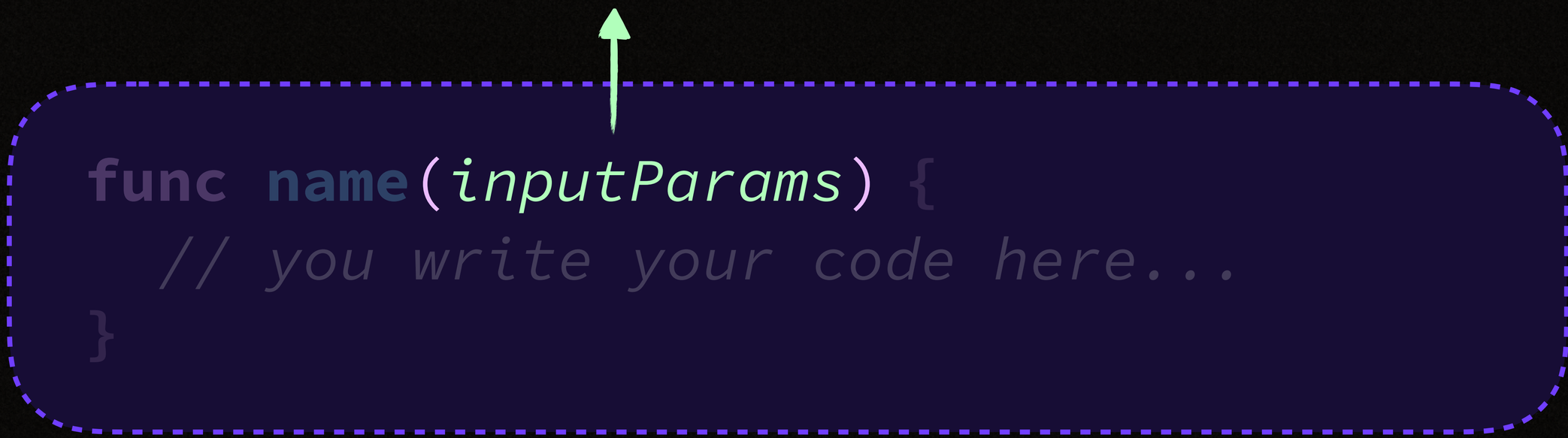
```
func sum(a int, b int) {}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

A function *may* declare *zero or more* input parameters

*(name type, name type)*



```
func name(inputParams) {
  // you write your code here...
}
```

```
func sum(a int, b int) {}   =   func sum(a, b int) {}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

"return" causes a function to stop and terminate immediately

```
func name(inputParams) {
  // you write your code here...
}
```

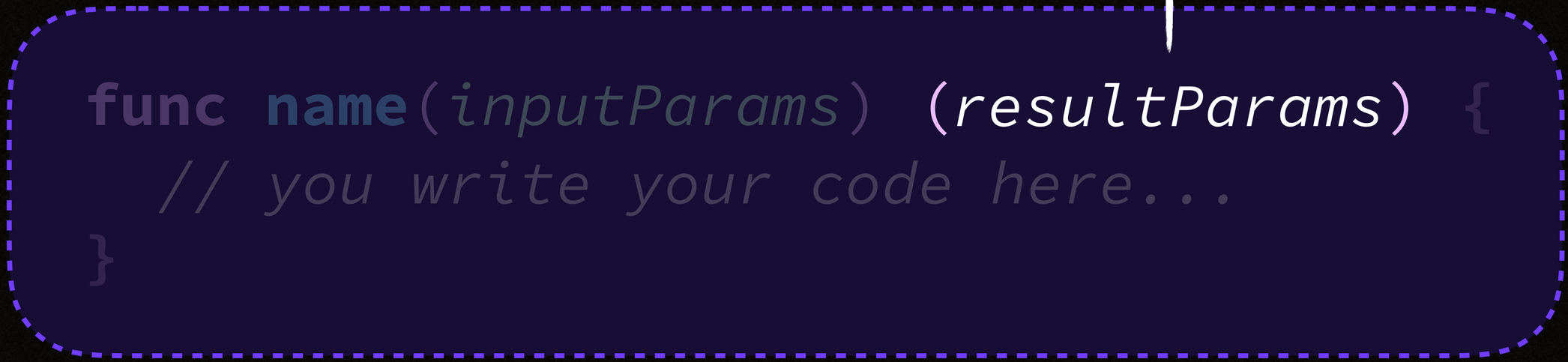
```
func Print() {
  // runs this code...
  return
  // skips running the rest...
}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

## A function *may* declare *zero or more* result values

*(type, type)*



```
func name(inputParams) (resultParams) {
  // you write your code here...
}
```

```
func Atoi(s string) (int, error) {}
```

👉 *Always declare an* ***error result value*** *as the* ***last parameter***

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

A function *may* declare *zero or more* result values

*(type, type)*

```
func name(inputParams) (resultParams) {
  // you write your code here...
}
```

```
func Atoi(s string) (int, error) {
  // code to convert s to int
  return 0, nil
}
```

👉 *Returned values will be **copied** to the calling function*

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

You *don't* need to use parentheses when a function declares a single result value

```
func name(inputParams) singleParam {
  // you write your code here...
}
```

```
func Itoa(i int) string {
  // code to convert i to string
  return "some string value"
}
```

👉 *You don't need to use a **comma** when **returning** **a single value***

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

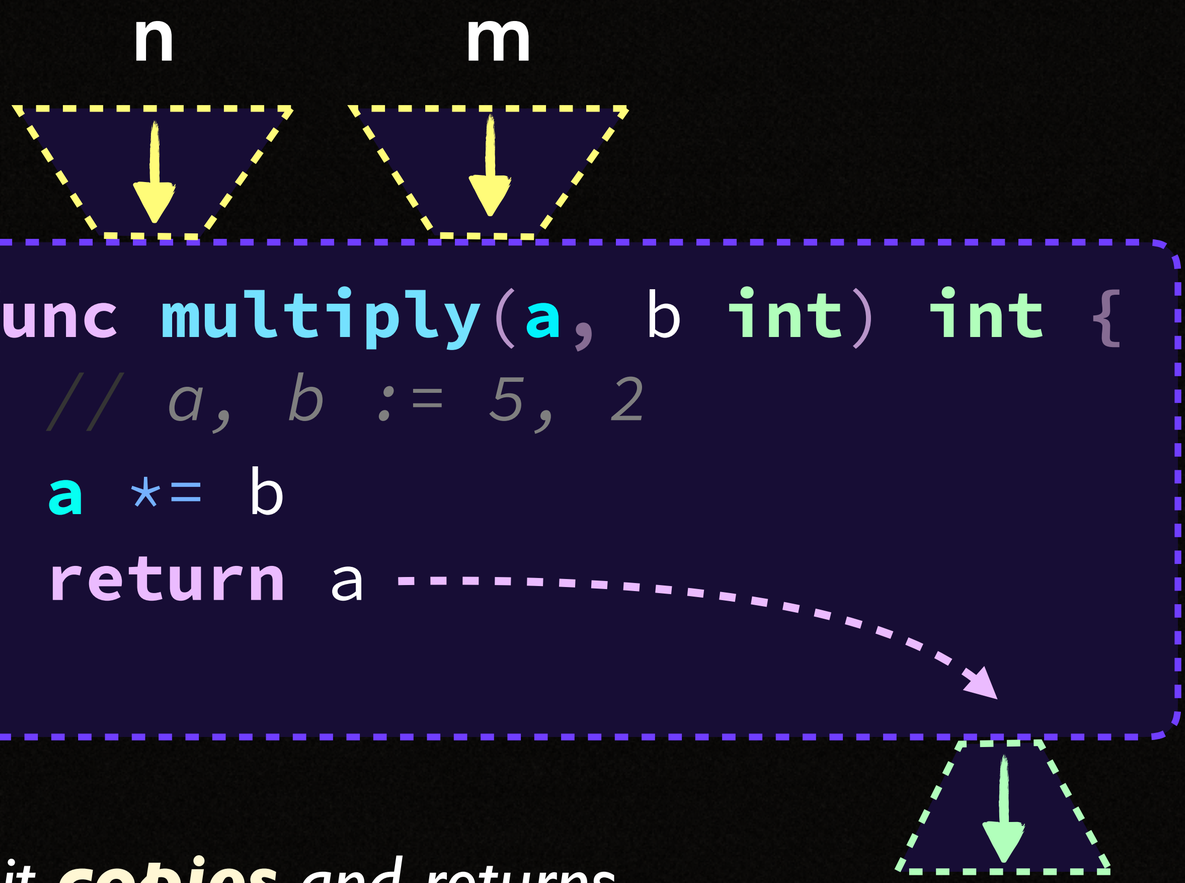
# FUNCTIONS

Input params and result values are copied

```
n, m := 5, 2
res := multiply(n, m)
// res ⟶ 10
// n   ⟶ 5
// m   ⟶ 2
```

**multiply** *cannot change* **n** *or* **m**

*it can only change* **a** *or* **b**




*it* **copies** *and returns the local variable:* **a**

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# BASICS — PART II

## Confine the variables to a function

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# FUNCTIONS

Package level variables can cause hard to debug problems later on

```
func incrN() {
  N++
}
```

```
var N int
```

```
func showN() {
  // ...
  fmt.Printf("N = %d\n", N)
}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

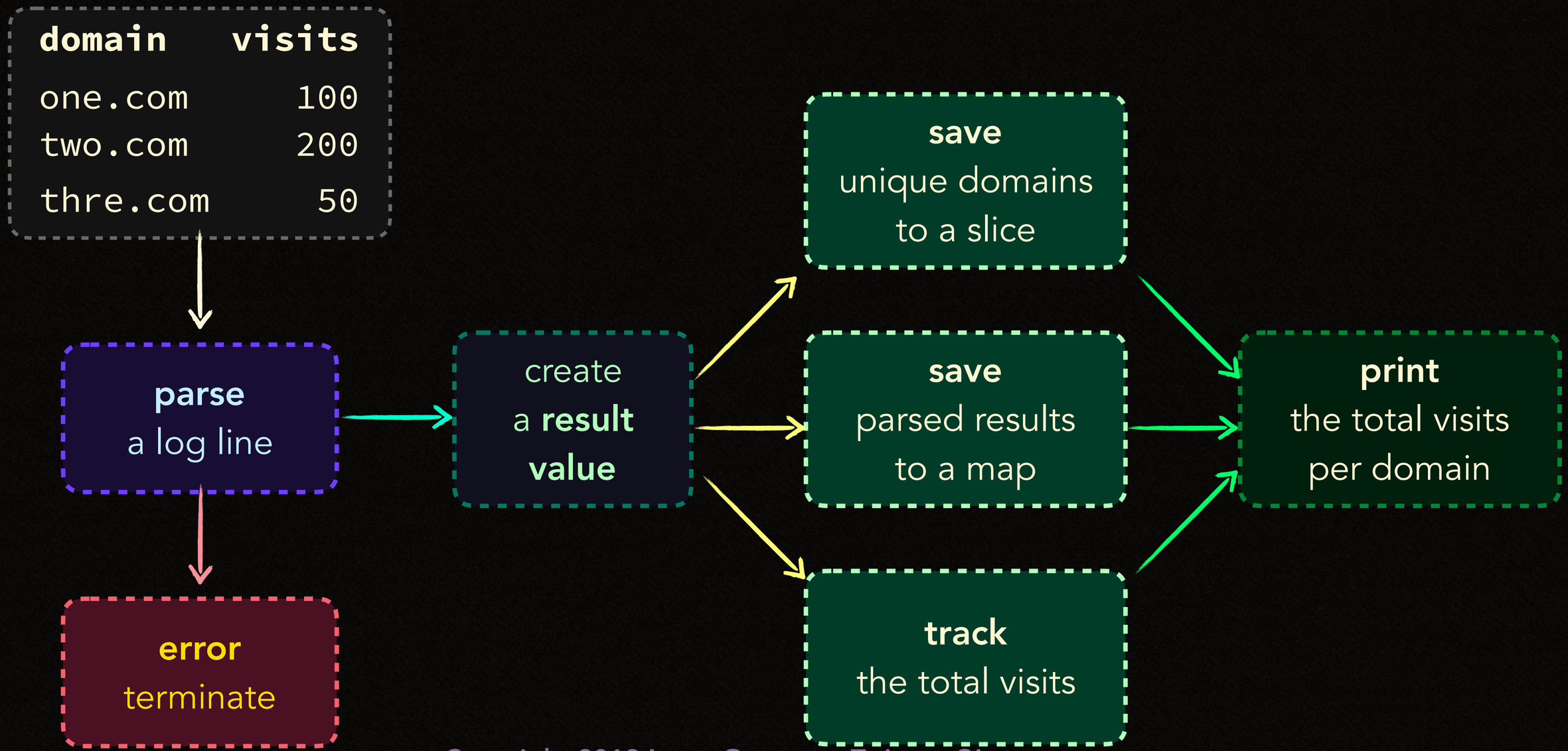
# LOG PARSER

**Let's rewrite the log parser program using functions!**

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# PROGRAM DESIGN

## Use verbs to find out the unique responsibilities = *functions*




Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# NAKED RETURN

Returns the named result values automatically

*The names of the result values*

```
func parse(p parser, line string) (parsed result, err error) {
    // ...
    var parsed result
    var err error

    return
}
```

⟵ `return parsed, err`

*Inside the function,*
the result values become variables

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

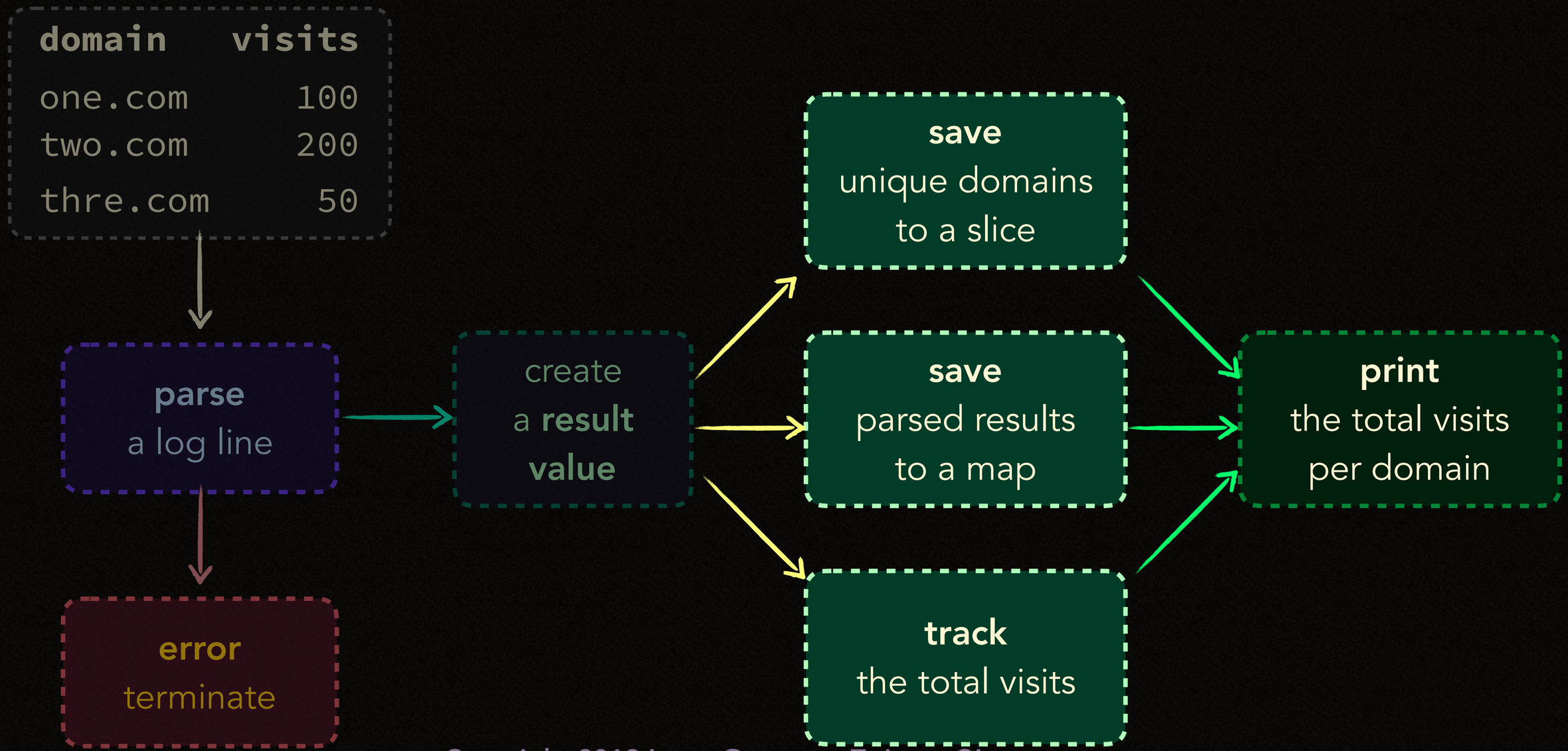
# PASS-BY-VALUE

**Everything is a copy!**

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# PROGRAM DESIGN

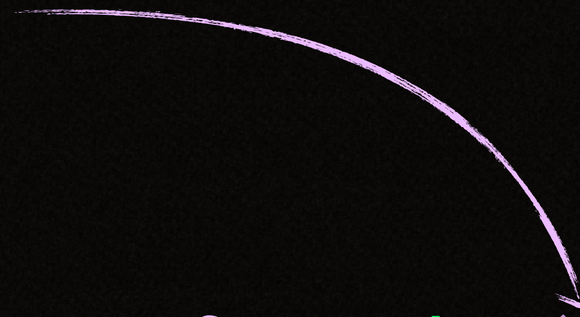
## Use verbs to find out the unique responsibilities = *functions*




Copyright 2019 Inanc Gumus — Twitter: @inancgumus

★ In Go, every value is copied when assigned to a variable or passed to a function ★

```
// main function's parser
parser {
   sum     : map[]
   domains: []
   total   : 0
}
```



```
func update(p parser, r result)
```

★ COPY ★

```
// update function's parser
parser {
   sum     : map[]
   domains: []
   total   : 0
}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

★ update() can only change its local copy of the parser struct value ★

```
// main function's parser
parser {
   sum     : map[]
   domains: []
   total   : 0
}
```

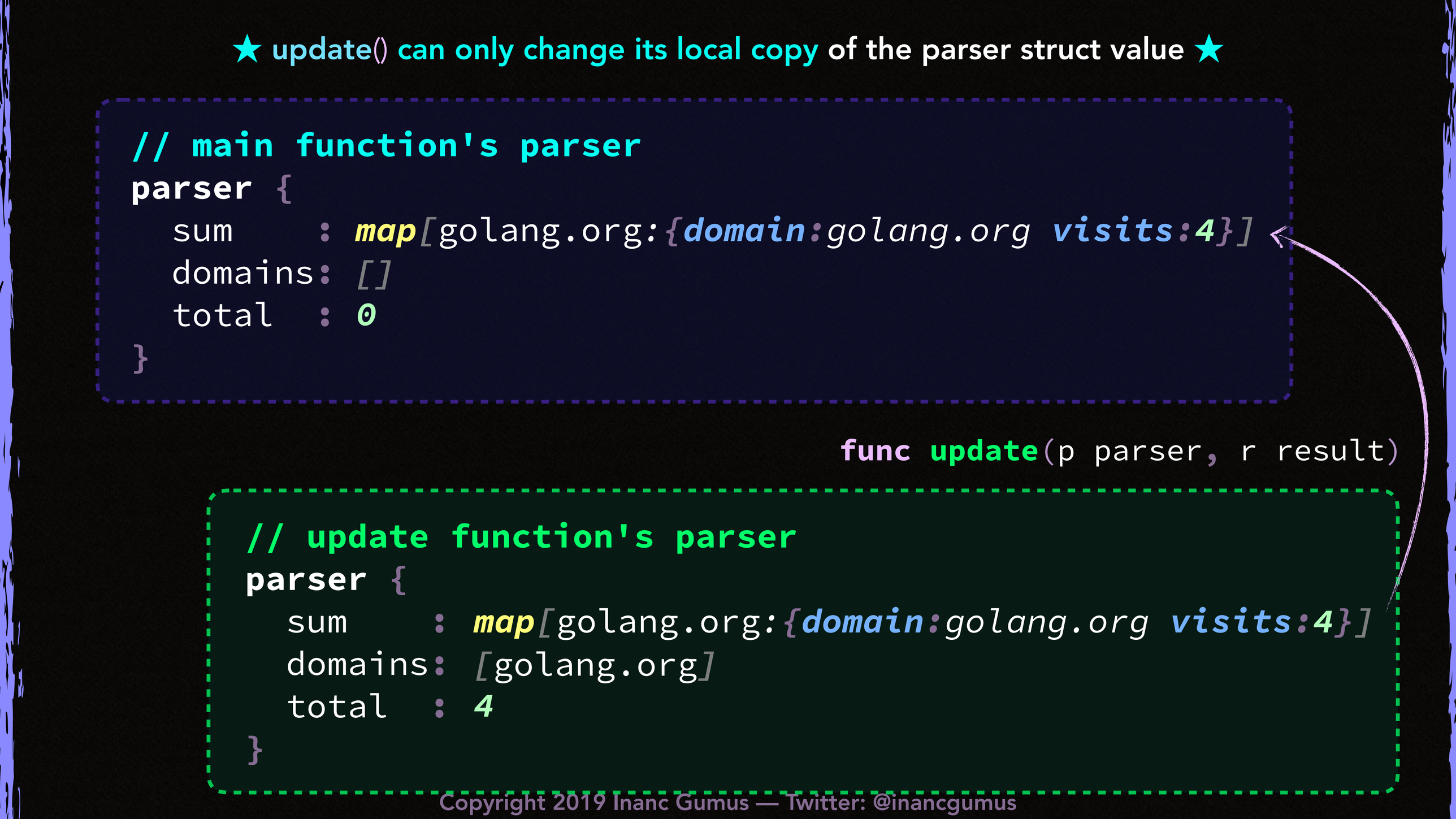
```
func update(p parser, r result)
```

```
// update function's parser
parser {
   sum     : map[golang.org:{domain:golang.org visits:4}]
   domains: [golang.org]
   total   : 4
}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

★ update() can only change its local copy of the parser struct value ★

```
// main function's parser
parser {
  sum    : map[golang.org:{domain:golang.org visits:4}]
  domains: []
  total  : 0
}
```

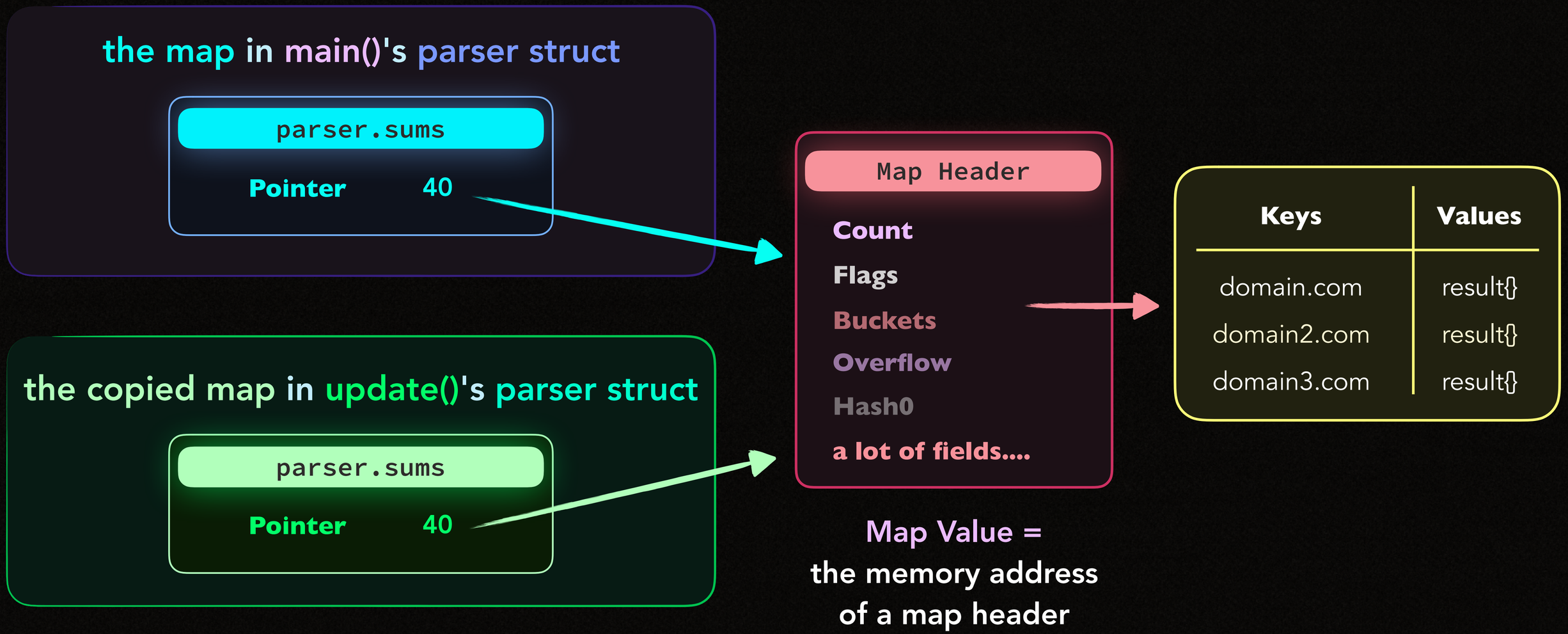


```
func update(p parser, r result)
```

```
// update function's parser
parser {
  sum    : map[golang.org:{domain:golang.org visits:4}]
  domains: [golang.org]
  total  : 4
}
```

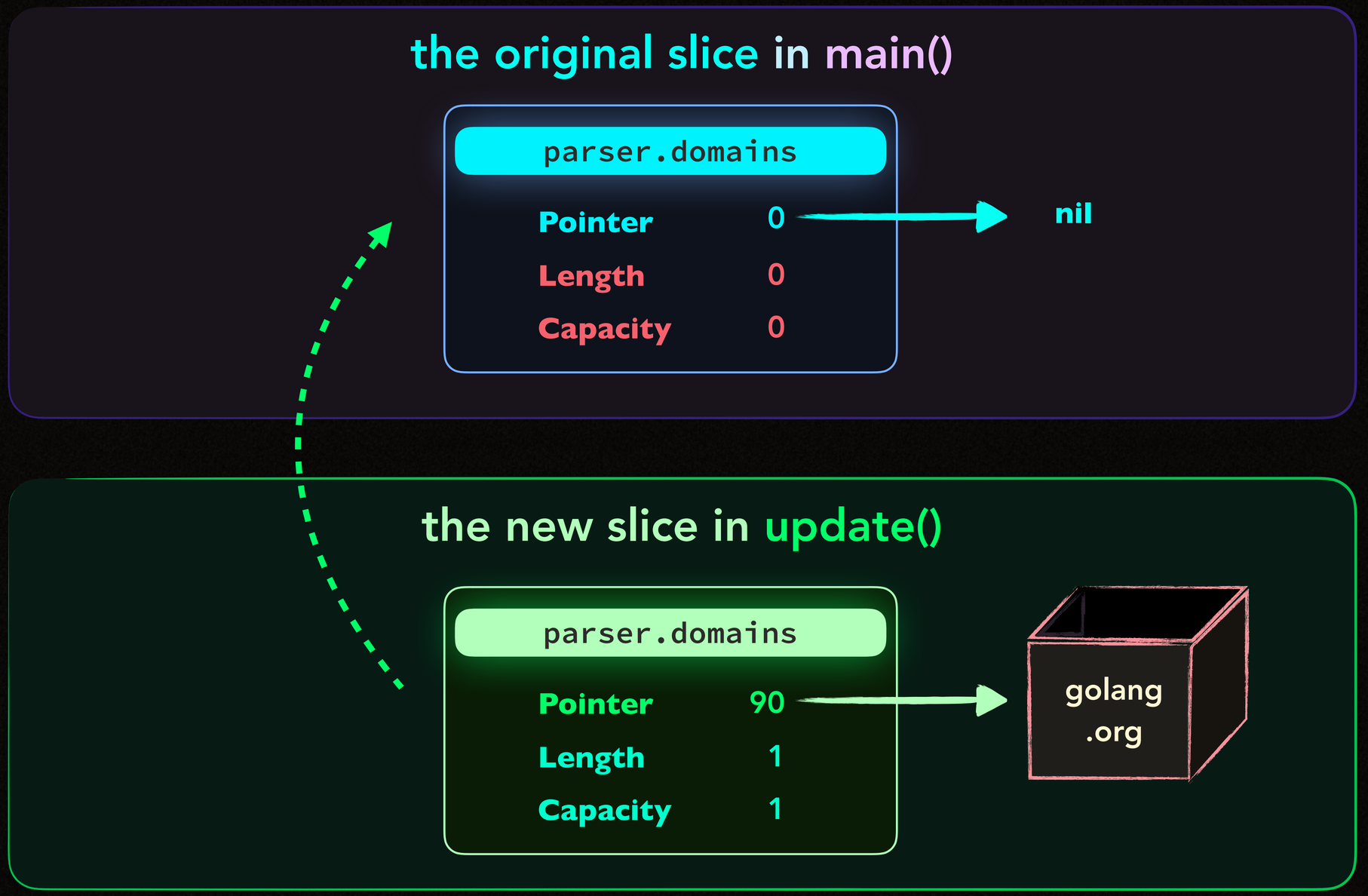
Copyright 2019 Inanc Gumus — Twitter: @inancgumus

When you copy a map value:
It will continue to point to the same memory location




Copyright 2019 Inanc Gumus — Twitter: @inancgumus

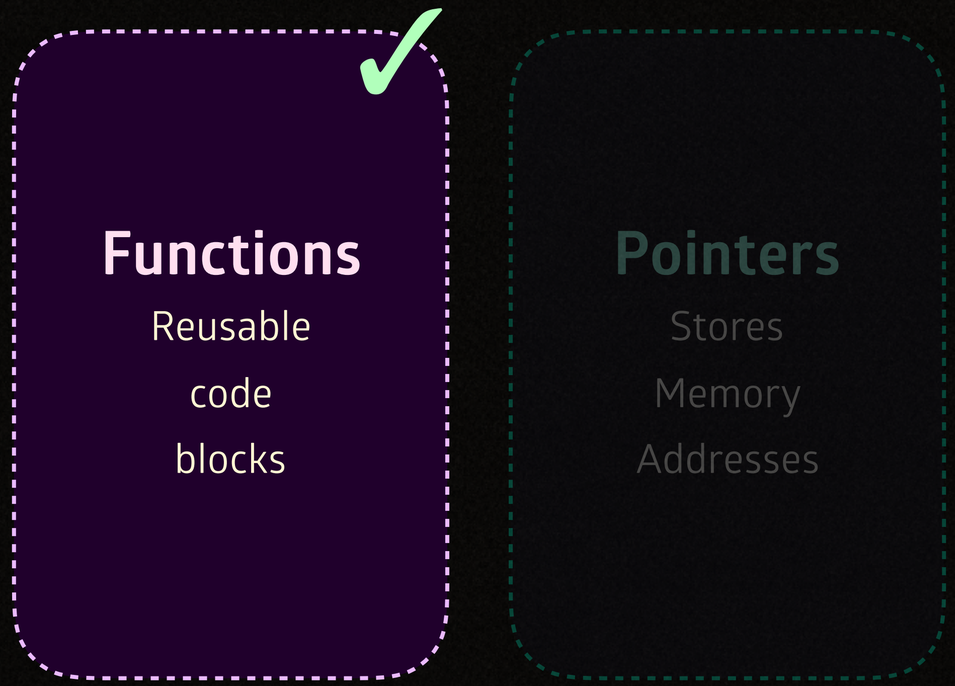
# When you change the copy of a slice value (*slice header*):
# You need to return it back




Copyright 2019 Inanc Gumus — Twitter: @inancgumus

# PART V

## Functions and Pointers




Copyright 2019 Inanc Gumus — Twitter: @inancgumus